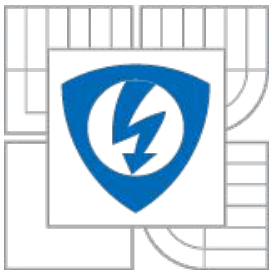
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

KALIBRACE KAMERY NA ZÁKLADĚ PŘÍMKOVÝCH ÚSEKŮ

CAMERA CALIBRATION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — JAN OPRAVIL
AUTHOR

VEDOUCÍ PRÁCE — ING. MILOSLAV RICHTER, PH.D.
SUPERVISOR

BRNO 2010

Abstrakt

Nedokonalost čoček některých průmyslových kamer působí rušivě. Tato práce se zabývá právě kalibrací kamer, které mají geometrické zkreslení typu soudek nebo poduška. Ke kalibraci se využívá funkce aproximující zkreslení reálné čočky. K výpočtu koeficientů této funkce se využívají nalezené zkreslené přímkové úseky ve snímku. Klíčovou roli hrají právě detekované přímky. Na ně je kladen požadavek, aby nesly co největší informaci o zkreslení. K jejich kvalitnější detekci se vhodně volí některé metody předzpracování. Vytvořené algoritmy jsou testovány nejprve na uměle generovaných a později i na reálných kalibrech. Celý program je psán v jazyku C a pro svoji činnost využívá některé funkce z knihovny openCV.

Klíčová slova: kalibrace kamery, geometrické zkreslení, objektiv, detekce přímek, obraz.

Abstract

Imperfection of some lenses used in industrial cameras is distracting. This thesis is concerned with the calibration of cameras with geometric barrel- or cushion-type distortion. In the calibrating process is used a function which approximates real lens distortion. To calculate the coefficients of this function the straight line sections in the distorted image are used. Those detected lines play the key role and they ought to have the greatest possible information bias. To enhance the quality of detection are chosen some methods of pre-processing. The algorithms thus created are tested first on an artificially generated calibers and later on real ones. The program is written in C and is using some of the functions of the OpenCV library.

Keywords: calibration of camera, geometric dictortion, lens, line detection, image.

OPRAVIL J.:*Kalibrace kamery na základě přímkových* úseků. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. Počet stran 56. Vedoucí bakalářské práce Ing. Miloslav Richter, Ph.D.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Kalibrace kamery na základě přímkových úseků jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne: 31. května 2010 ……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Miloslav Richter, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

Obsah

1.Úvod

Počítačové vidění se začíná prosazovat v mnoha oborech. Například pokud projedeme na křižovatce na červenou, můžeme být vyfoceni kamerou, která díky počítačovému vidění rozpozná jednotlivé znaky naší SPZ a takto může být vozidlo snadno identifikováno.

Ve většině případů kamera snímá obraz, který je dál zpracován algoritmem pro konkrétní úlohu. Obraz snímaný například průmyslovou kamerou může mít hned několik nedostatků. Jedním z nich je i geometrické zkreslení, které je způsobeno nedokonalostí optické čočky kamery.

Zkreslení působí v obraze rušivě a je potřeba ho co nejlépe odstranit. Tato práce se bude tedy zabývat právě kalibrací zkreslení kamer. Nejtypičtější pro objektivy je soudkové zkreslení. Pro jeho kalibraci využijeme přímkové úseky nalezené ve zkresleném obraze.

Základem kalibrace je správné vyhledání zkreslených přímek nebo jejich úseků v obraze. Poté nalezení koeficientů degradační funkce, pomocí které dojde k vyrovnání zkreslených přímek a tedy i celého obrazu. Nalezené koeficienty zkreslení jsou charakteristické pro daný objektiv a tudíž budou u všech snímků stejné.

Před samotným vyhledáním přímek musíme obraz nejdříve předzpracovat. Možností předzpracování obrazu je celá řada, základní z nich si popíšeme v následujících kapitolách. Jedním z hlavních úloh předzpracování bude detekce hran, z které pak vychází detekce přímek potřebných ke kalibraci kamery.

Přímky nebo jejich úseky musíme detekovat tak, aby měly co nejvyšší vypovídací schopnost o zkreslení obrazu. Přímky se budeme pokoušet vyrovnat pomocí inverzní funkce k funkci, která obraz zkreslila. Celý program vyzkoušíme na řadě připravených snímků.

Program budeme psát v jazyku C, za podpory knihovny openCV. Naší prioritou nebude optimalizace výpočtů, ale rozpracování algoritmů pro plnění výše zmíněných úkolů.

2.Počítačové vidění

Jedním z nejdůležitějších smyslů člověka je zrak, pomocí kterého se člověk orientuje, rozpoznává objekty. Lidský mozek zpracovává zrakové informace, vybírá z nich ty důležité, které pak vnímáme. Počítačové vidění je obor, který se snaží napodobit lidské vidění.

Vlastností počítačového vidění se hojně využívá v mnoha oblastech, ve kterých nemá zastoupení. Například v medicíně, kde se využívjí vlastnosti zobrazení elektromagnetického záření mimo oblast viditelnou lidským okem. Viditelná část spektra pro lidské oko je přibližně 380 až 720 nm. V tomto intervalu vnímáme záření s určitou vlnovou délkou jako barvu. Elektromagnetické záření s kratší vlnovou délkou (což znamená s vyšší frekvencí) nese větší energii, kterou vnímáme jako fialovou barvu, naopak viditelné záření s malou energií vnímáme jako červenou barvu. Kombinací elektromagnetického záření o různých vlnových délkách vznikají barvy.

Pokud budeme uvažovat dvourozměrný statický nepohyblivý obraz, který vnímá lidské oko, můžeme ho matematicky popsat spojitou obrazovou funkcí $z=f(x,y)$. Obor hodnot funkce z je v rozsahu viditelného záření. Reálná čísla x ,y jsou souřadnice bodu.

Pro oblast počítačového vidění je obrazová funkce prakticky vždy diskretizována. Proto pro stejný obraz si lze diskretizovanou obrazovou funkci z představit jako množinu jednotlivých bodů, tyto body nazýváme pixely (z anglické zkratky *picture element*). Množina pixelů je rastr, který je definičním oborem diskretizované funkce z. Obor hodnot diskretizované funkce z bude nabývat diskrétních hodnot. Tento proces se nazývá digitalizace.

2.1.Digitalizace

Digitalizace je základní děj počítačového vidění. Jeho hlavním úkolem je převedení spojité obrazové funkce z na její diskrétní ekvivalent I. Digitalizace převede jak definiční obor D, tak obor hodnot OH. Setkáváme se s ní v každodenním životě, jako příklad poslouží jakýkoliv digitální fotoaparát, digitální kamera. Spojitý obraz, který snímáme na fotoaparát, má teoreticky nekonečný

rozsah obrazových hodnot, stejně jako nekonečné rozlišení. Tento spojitý obraz po digitalizaci bude mít omezené rozlišení a konečný počet barev. U digitalizace tedy nutně dochází ke ztrátě informace. Nelze tedy z diskrétní funkce *I* získat zpět stejnou spojitou funkci *z,* z které jsme vycházeli.

Digitalizace probíhá ve dvou hlavních krocích: kvantování a vzorkování.

2.1.1.Kvantování

Kvantování přiřadí diskretizované funkci *I* nový diskrétní *OH*. Tento nový *OH* nabývá, v případě počítačového zpracování celých čísel, nejčastěji 2^b. Kde *b* je počet kvantizačních bitů. Velmi často je *b* rovno 8, *OH* tedy má 256 kvantizačních úrovní, které představují jas ve spojitém obraze. Tyto úrovně mohou mít konstantní nebo nerovnoměrnou délku intervalu. Výhoda nerovnoměrné délky kvantizačního intervalu spočívá v částečném potlačení kvantizační chyby. Častěji se však z důvodu jednoduchosti používá konstantní.

Kvantizační chyba je obsažena v digitálním obraze ve větší či menší míře vždy. Pokud je použito menší množství kvantizačních úrovní, než je lidské oko schopno rozlišit, vzniká v obraze s původně hladkým přechodem barvy, přechod skokový. Lidské oko tento jev vnímá jako rušivé hrany.

2.1.2.Vzorkování

Vzorkování změní definiční obor funkce *z* na nový celočíselný *D*, to znamená, že převede spojitou funkci *z* do matice *M*. V tomto procesu dochází ke ztrátě informace, proto čím větších rozměrů nabývá matice *M*, tím přesněji je spojitý obraz aproximován. V praxi se používá pravidelná vzorkovací mřížka. Nejčastěji čtvercová, díky snadnější technické realizaci. Může se ale využít i hexagonální a trojúhelníková. Interval vzorkování musíme volit tak, aby nedocházelo ke zkreslení, Shannonův teorém říká, že interval musí být stejný nebo nejlépe menší, než polovina rozměru nejmenších detailů v obraze.

2.2.Barvy

Jak bylo popsáno v kapitole (2.1.1.), lidské oko vnímá elektromagnetické záření o různé vlnové délce jako barvu. V počítačové grafice zavádíme model (barevné prostory), který popisuje barvy pro jejich rekonstrukci. Modely jsou tvořeny několika základními barvami. Těchto modelů je celá řada, liší se počtem a výběrem základních barev, výpočtem výsledné barvy atd.

V praxi se nejčastěji používají následující:

- RGB;
- CMY, CMYK;
- HSV a HLS;
- Televizní YUV;
- Chromatický diagram CIE.

2.3.Předzpracování

Cílem je zdůraznění či potlačení určitých charakteristik obrazu pro další zpracování. Je zřejmé, že pro každou úlohu zpracování obrazu je nutné použít odlišný způsob předzpracování. Předzpracování se nejčastěji používá na:

- potlačení šumu;
- transformaci jasové úrovně;
- ostření a detekce hran;
- morfologie;
- transformaci barev;
- zvýšení kontrastu;
- obnovení obrazu při známé degradaci.

2.3.1.Potlačení šumu

Šum, až na výjimky, je nepotřebná rušivá informace, která se přidala k původnímu obrazu digitalizací nebo kompresí.

Plně odstranit šum bychom dokázali pouze za předpokladu, že bychom měli k dispozici jeho funkci, tu ale mít v praxi nikdy nebudeme. Proto si uvedeme některé metody potlačující šum. Každá metoda je vhodná na jiný typ šumu a na jiná další zpracování.

2.3.1.a) Průměrování

Realizuje nahrazení původní hodnoty jasu průměrnou hodnotou jeho okolí. Okolí je dáno konvoluční maskou. Čím větší maska bude, tím více poklesne šum, ale za cenu většího rozmazání hran. Maximální velikost masky je dána nejmenším detailem v obraze, který chceme zachovat nerozmazaný.

Pro odstranění šumu můžeme také použít vážené průměrování. Váhy jsou dány jednotlivými koeficienty masky. Typicky se zvětšují váhy středového bodu nebo jeho čtyř sousedů. Tato metoda je úspěšná při Gaussově šumu. Součet hodnot v masce by měl být roven jedné, aby filtr nezesvětloval, respektive neztmavoval výsledný obrázek. To platí u všech metod, které potlačují šum. Úkázka nejpoužívanějších masek je na obrázku 1.

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}_A \qquad \frac{1}{16}\begin{bmatrix}1 & 1 & 1\\ 2 & 4 & 2\\ 1 & 1 & 1\end{bmatrix}_B$$

Obrázek 1: Masky pro průměrování.

2.3.1.b) Gaussův filtr

Gaussův filtr je rozšíření metody průměrování s tím rozdílem, že hodnoty v konvoluční masce jsou dány Gaussovou funkcí vzorec č.1. Tato metoda vede k rozmazání obrázku, což může být problémem pro další zpracování. Gaussův filtr je efektní k potlačení Gaussova šumu.

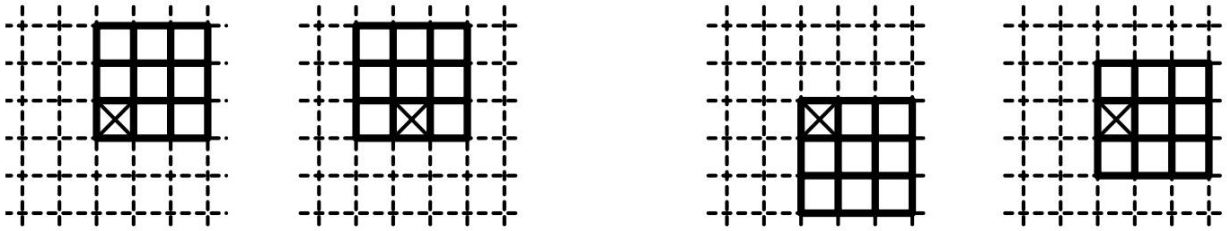
$$G(x,y)=\frac{1}{2\cdot pi\cdot\rho}\cdot e^{-\frac{x^2+y^2}{2\cdot\rho^2}}$$ Vzorec 1: Gaussovo rozložení

2.3.1.c) Filtrace metodou mediánu

Metoda stanoví jas výsledného bodu jako medián určený velikostí masky. Medián určíme pro diskrétní obrazovou funkci tak, že seřadíme vzestupně okolní hodnoty jasu a medián je uprostřed tohoto výčtu. Tato metoda dobře potlačuje šum, ale narušuje tenké čáry a ostré rohy v obraze. Proto se využívá masek tvaru kříže, nebo písmene X častěji než čtvercových masek. Pokud použijeme masku ve tvaru kříže, neporušíme diagonální hrany, ale pokud se rozhodneme pro masku ve tvaru X, zůstanou zachovány vodorovné a svislé hrany. Metoda dobře filtruje převážně šum typu pepř a sůl (impulzní šum).

2.3.1.d) Rotující maska

Tato metoda se pokouší najít příbuzné okolí, které má podobný jas jako ostřený bod. Pro výpočet výsledné hodnoty jasu daného bodu je použit průměr okolí, které nejvíce odpovídá původnímu bodu. Jako okolí bodu můžeme použít libovolnou masku, která bude rotovat kolem daného bodu. Je zřejmé, že čím menší masku použijeme, tím menší jsou změny v obraze. Na obrázku 2 je znázorněna rotující maska kolem jednoho bodu.



Obrázek 2: Rotace masky kolem bodu (Hlaváč, Šonka, 1992).

Metoda tedy nerozmazává hrany a má mírně ostřící charakter,. Šum odstraňuje velmi dobře, ale vytváří plošky se stejnou hodnotou jasu. Pokud se filtrují předměty vytvořené člověkem, nepůsobí tyto plošky rušivě. Proto se tato metoda používá nejčastěji.

2.3.2.Ostření a detekce hran

Jak uvádí *Žára, Beneš, Sochor a Felkel (2004)* lidské vnímání je založeno na rozpoznávání hran. Jednou z možností jak zvýraznit nějaký obraz, abychom ho vnímali jako ostřejší, je zvýraznit v něm hrany. Hranu v diskrétním obraze vnímáme tam, kde dochází k výrazné změně sousedních pixelů. Hrana je vysokofrekvenční informace, a proto je její zvýraznění inverzní operací k odstranění šumu.

Hrana je určena velikostí a směrem. Je tedy vektorovou veličinou, která vychází z gradientu, popsaného operátorem ∇. Velikost gradientu se vypočítá jako velikost vektorů parciálních derivací obrazové funkce, jak ukazuje rovnice 2. Směr gradientu popisuje operátor Ψ a vypočítá se pomocí rovnice 3.

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\delta f(x,y)}{\delta x}\right)^2 + \left(\frac{\delta f(x,y)}{\delta y}\right)^2}$$

Vzorec 2: Velikost gradientu

$$\Psi = \tan^{-1} \frac{\frac{\delta f}{\delta y}}{\frac{\delta f}{\delta x}}$$ Vzorec 3: Směr gradientu

Výše uvedené vzorce platí pro spojité obrazové funkce. V diskrétním obraze aproximujeme parciální derivace diferencemi $\Delta_i g(i, j)$, $\Delta_j g(i, j)$, které vypočítáme podle vzorce 4 a 5.

$$\Delta_i g(i, j) = g(i, j) - g(i-n, j)$$ Vzorec 4: Diference v ose i

$$\Delta_i g(i, j) = g(i, j) - g(i-n, j)$$ Vzorec 5: Diference v ose j

Diferenci lze využít na ostření hran v obraze tak, jak ukazuje Vzorec 6: Ostření hran v obraze, kde *f(i,j)* je výsledek ostření, *g(i,j)* je vstupní obrazová funkce, *s(i,j)* je funkce udávající velikost gradientu obrazové funkce a parametr *c* je kladný součinitel udávající míru ostření.

Hranové operátory pracují buď na vyhledávání lokálních minim, respektive maxim první derivace (Sobelův a Robertsův operátor), nebo principu průchodu nulou druhé derivace (Laplaceův operátor) obrazové funkce .

$$f(i, j) = g(i, j) + c\, s(i, j)$$ Vzorec 6: Ostření hran v obraze

Operátory, které zvýrazňují hrany, zvýrazní všechny vysoké frekvence a tedy i šum. Méně zvýrazňují šum operátory, které používají k výpočtu diference větší okolí bodu.

Součet hodnot obsažených v konvoluční masce by měl být roven nule proto, aby v oblastech s konstantním jasem byla konvoluce nulová. Nejjednodušší hranový operátor je již zmíněný Robertsův. Velikost gradientu počítaná podle vzorce 7, používá okolí 2x2 pixelů, jeho konvoluční

maska je na obrázku 3. Robertsův operátor detekuje především hrany se sklonem 45°. Tento operátor tedy aproximuje první derivaci, je směrově závislý a velmi citlivý na šum.

Lepší aproximace první derivace používá Sobelův operátor, jeho konvoluční maska je na obrázku 4. Sobelův operátor je stejně jako Robertsův závislý na směru. Kompletní sadu masek získáme rotací původní masky o 45°.

$$|\delta g(i,j)| = |g(i,j) - g(i+1,j+1)| + |g(i,j+1) - g(i+1,j)|$$ Vzorec 7: Robertsův operátor

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

Obrázek 3: Robertsův operátor.

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Obrázek 4: Sobelův operátor.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}_A \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}_B$$

Obrázek 5: Laplaceův operátor.

Jak bylo již zmíněno, aproximaci druhé derivace využívá Laplaceův operátor. Konvoluční maska je na obrázku 5 a je invariantní vůči rotaci. Laplaceův operátor je více náchylný na šum, než operátory aproximující první derivaci. Výběr vhodného gradientního operátoru závisí na charakteru vstupního obrazu a na konkrétní úloze.

2.3.3.Morfologie

Metody matematické morfologie se používají především pro:

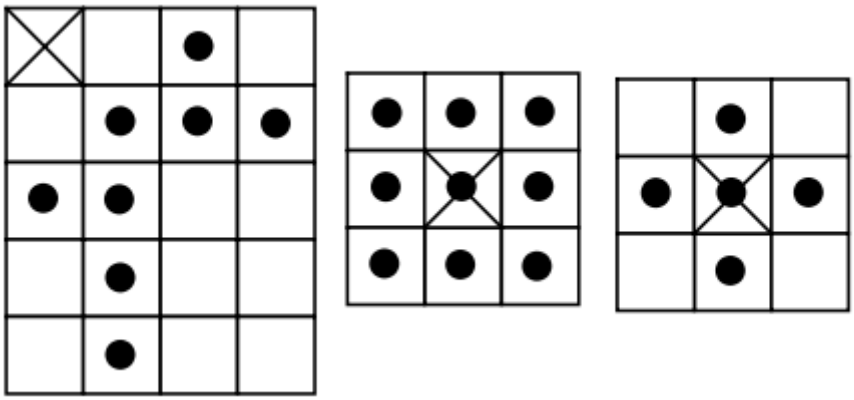
- odstranění šumu, zjednodušení tvaru objektů;
- zdůraznění struktury objektů (kostra, ztenčování, značkování objektů, atd.);
- popis objektů číselnými charakteristikami (plocha, obvod, projekce, atd.).

Matematická morfologie vychází z vlastností bodových množin. Lze zpracovávat jak obrazy s více úrovněmi jasu, tak i binární obrazy. Prakticky používané morfologické transformace jsou realizovány jako relace obrazu s jinou menší bodovou množinou, které se říká strukturní element. Příklad strukturních elementů je na obrázku č. 6. (*Hlaváč, Šonka,1992*)

Morfologickou transformaci si můžeme představit jako systematický pohyb strukturního elementu po obraze.

Matematická morfologie se používá nejčastěji pro:

- dilataci a erozi;
- otevření a uzavření;
- skelet.



Obrázek 6: Příklady strukturních elementů.

2.3.3.a) Dilatace

Skládá body dvou množin pomocí vektorového součtu (Minkowského množinový součet). Dilatace rozšiřuje objekt o jednu skupinu na úkor pozadí. Pokud byly v objektu díry s tloušťkou jednoho pixelu, tak je tato operace zaplnila.

2.3.3.b) Eroze

Skládá body dvou množin pomocí vektorového rozdílu, je to duální transformace k dilataci. Objekty které mají tloušťku jednoho pixelu zmizí. Eroze se používá ke zjednodušení objektů. Také se eroze používá, pokud chceme najít obrys objektů v obraze, potom stačí odečíst od původního obrazu jeho erozi.

2.3.3.c) Otevření a uzavření

Tyto transformace se používají k odstranění nadbytečných detailů v obraze, které jsou menší než strukturní element. Objekt tedy zůstane neporušen.

Uzavření spojí objekty, které jsou blízko u sebe a zaplní malé díry. Otevření oddělí objekty, spojené úzkými čarami. Jaké objekty budou spojeny nebo odděleny, záleží na velikosti a tvaru strukturního elementu. (*Hlaváč, Šonka, 1992*)

2.4.Zpracování obrazu

Je to velmi široký pojem, který závisí na konkrétním požadavku. Zpracování obrazu je jednou z nejobtížnějších, ale také nejdůležitějších částí. Zabývá se analýzou vstupního obrazu. Použité aplikace mohou být například hledání, počítání a měření objektů v obraze. Zpracování obrazu se nejčastěji provádí segmentací.

Segmentace obrazu je proces, kterým rozdělíme obraz do oddělených množin. Každá množina odpovídá objektu nebo oblasti s určitými vlastnostmi. Pokud hovoříme o kompletní segmentaci máme na mysli skupinu segmentovaných objektů, které mají přesnou shodu

s požadovaným objektem. Pokud segmentovaný objekt nesouhlasí, pak jde o částečnou segmentaci. Hlavní problémy segmentace jsou nejednoznačnost obrazových dat a šum.

Základní segmentační metody jsou tyto:

- segmentace prahováním;
- segmentace na základě detekce hran;
- segmentace pomocí Hougovy transformace.

2.4.1.Segmentace prahováním

Je to nejstarší a nejjednodušší metoda segmentace. Pokud budeme předpokládat, že segmentovaný objekt má na celé své ploše téměř konstantní jas, který je rozdílný od konstantního jasu pozadí, pak při vhodné volbě prahové úrovně je metoda velmi spolehlivá. Vzhledem k nenáročnosti výpočtu, který spočívá pouze v porovnání jasové úrovně, jak vyplývá ze vztahu 8, je tato metoda použitelná i v reálném čase. Výstupem této metody je binární obraz.

$$g(i,j)=\begin{matrix}1\ pro\ f(i,j)\geq T\\ 0\ pro\ f(i,j)<T\end{matrix}$$

Vzorec 8: Segmentace prahováním

T - volba prahu

V mnoha případech však segmentovaný objekt má proměnný jas (například z důvodu nerovnoměrností osvětlení). V tomto případě musíme použít proměnný práh jasové úrovně. Hodnota proměnného prahu je určována z tvaru histogramu.

Jsou-li v obraze objekty přibližně téhož jasu jasově odlišné od pozadí, je histogram jasu dvouvrcholový. Jeden vrchol odpovídá četnosti obrazových elementů pozadí, druhý četnosti obrazových elementů objektů. Z tvaru histogramu je zřejmé, že hodnoty jasu ležící mezi oběma vrcholy nejsou v obraze časté a odpovídají jasu hraničních obrazových elementů mezi objekty a pozadím. Výsledný práh určíme jako minimum, které leží mezi dvěma maximy. (*Hlaváč, Šonka,1992*)

Velkým problémem zmíněného postupu je předpoklad minimálního zašumění obrazu. Pokud je vstupní obraz zašuměn, také histogram bude zašuměn. Pro odstranění zašumění musíme

použít některou metodu z kapitoly 2.3.1. nebo alespoň metodu na vyhlazení histogramu (například konvolucí).

Další modifikací je prahování s více prahy, kdy výsledkem již není binární obraz, ale obraz s určitým počtem jasových úrovní. Postup určení prahu je obdobný s využitím vícevrcholového histogramu.

2.4.2.Segmentace na základě detekce hran

Hranice objektů tvoří hrany. Hrany můžeme detekovat pomocí hranových detektorů (kapitola 2.3.2.). Hranové detektory naleznou místa, v nichž dochází k určité nespojitosti (změna jasu, barvy, textury, šum). Je tedy zřejmé, že ne všechny nalezené hrany odpovídají skutečným hranám segmentovaného objektu. Zásadní problém je tedy výskyt hran bez přítomnosti skutečné hranice, anebo absence hran tam, kde existuje skutečná hranice. Výstup z hranových detektorů se tedy musí dále zpracovávat.

2.4.2.a) Sledování hranice

Tato metoda se používá v případě, pokud neznáme tvar hranice. Procházíme obraz po řádcích, dokud nenalezneme obrazový bod. Ten náleží hranici objektu (aktuální bod). Poté vyšetřujeme okolí 3x3 bodů s aktuálním bodem uprostřed. Z osmi okolních bodů vybereme bod, který má maximální úroveň šedé. Tento bod také tvoří hranici objektu. Iterativním postupem zajišťujeme další body, které tvoří hranici. Jako následující body vždy uvažujeme bod, který leží symetricky k poslednímu bodu vzhledem k aktuálnímu bodu.

2.4.2.b) Segmentace narůstáním oblastí

Tato metoda vyhledává množinu obrazových bodů se stejnou (nebo přibližně stejnou) vlastností, na základě které segmentujeme (úroveň šedé). Začneme s určitým bodem a zkoumáme okolní body. Pokud mají stejnou hodnotu jasu, jsou spojeny do oblastí. Při této metodě se oblasti rozrůstají s počátečním stavem jednoho obrazového bodu. Výsledkem je mnoho oblastí. K těmto

oblastem přidáme soubor parametrů (např. jasovou úroveň). Následuje zkoumání všech hranic mezi sousedními oblastmi.

Je počítána „velikost“ hranice z rozdílů průměrných vlastností sousedních oblastí. Hranice je velká, pokud vlastnosti v sousedních oblastech jsou značně odlišné. V opačném případě je hranice malá. Velké hranice jsou ponechány, malé hranice se vypustí a sousední hranice splynou. Slučování sousedních oblastí pokračuje, až jsou odstraněny všechny malé hranice. V této etapě je dokončena segmentace.

Tato metoda je zpravidla náročná na strojový čas. Na druhé straně je dost účinná při segmentaci přirozených scén, o kterých nemáme předběžnou znalost. (*Z. Sobotka, M. Sobotka, 1990*) Tuto metodu lze uplatnit v obrazech se šumem, kde se obtížně hledají hranice.

2.4.3.Segmentace pomocí Hougovy transformace

Hugova transformace je obecná metoda sloužící ke hledání definovaných objektů, které jsou parametricky popsány. Klasická Hougova transformace slouží především k detekci úseček, kružnic a elips. Pokud se zaměříme na detekci přímek pomocí klasické Hougovy transformace, vyjádříme přímku pomocí parametrické rovnice 9, kde r je vzdálenost přímky od počátku (r je kladné číslo o maximální velikosti úhlopříčky obrazu) a úhel Φ svírá kolmice z počátku k dané přímce a osa x (Φ= <0-360).

Před aplikací transformace nadefinujeme nulovou matici, která bude reprezentovat akumulátor. Řádky matice budou představovat úhel Φ, a sloupce velikost r. Procházíme vstupní binární obrázek bod po bodu, dokud nenalezneme bod objektu (přímky). Poté dosadíme souřadnice x, y nalezeného bodu do rovnice 9 a vypočítáme r pro všechny úhly Φ. Pro dvojici takto získaných hodnot $[r,\Phi]$ přičteme jedničku do akumulační matice. Tento postup opakujeme, dokud neprojdeme celý obrázek. Po dokončení najdeme lokální maxima akumulační matice, která definují nalezenou přímku.

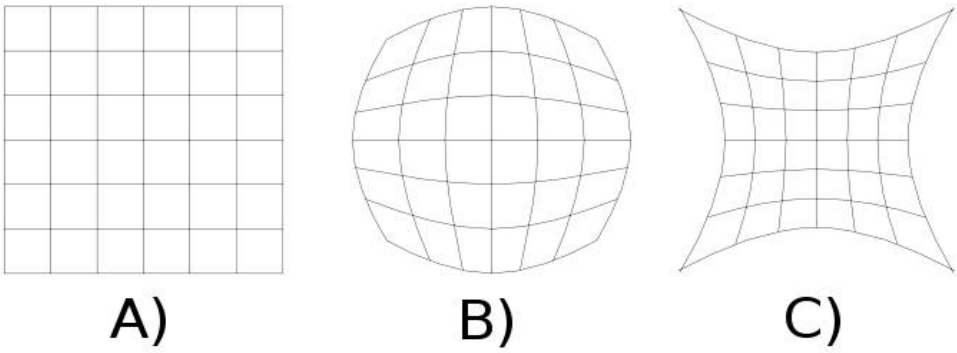
$$r = x\cos\theta + y\sin\theta$$ Vzorec 9: Parametrická rovnice přímky

2.4.4.Geometrická transformace, zkreslení

Základem transformace je předpis, který každému bodu ze vstupního obrazu, přiřadí jiný bod ve výstupním obrazu. Základní geometrické plošné transformace dělíme na:

- Lineární (posunutí, otočení, změna měřítka).
- Afinní (zkosení).
- Projektivní .
- Polynomické.

V této práci se budeme zabývat pouze zkreslením, které je nejčastější u kamer, tedy zkreslení typu soudek nebo poduška. Ty jsou znázorněny na obrázku 7.



Obrázek 7: Geometrické zkreslení, A) originál, B) soudek, C) poduška.

S touto transformací se poměrně často setkáváme při použití některých objektivů. Působí jako nežádoucí jev, tedy zkreslení. Způsob transformace vyjadřuje funkce 10. Vstupy funkcí *x (u, v)* a *y (u, v)* jsou souřadnice *u* a *v*, funkce udává novou polohu pixelu.

$$T(u,v)=[x(u,v),y(u,v)]$$

Vzorec10: Transformace

Pokud bychom chtěli na obraz aplikovat tuto transformaci, můžeme použít dvě metody mapování pixelů, to určuje způsob přiřazení pixelů z originálního obrazu do transformovaného.

- Dopředné mapování – prochází pixely vstupního obrazu a hledá jejich umístění ve výstupním obrazu. U tohoto mapováním mohou zůstat některé pixely ve výstupním obrazu volné.
- Zpětné mapování – je to opačná metoda, která prochází pixely výstupního obrazu a hledá k nim odpovídající ve vstupním obrazu. Problém může nastat, pokud je mapovaný pixel mimo vstupní obraz.

Transformace se většinou používá na korekci zkresleného obrazu, kdy se každý pixel posune radiálně. Posunutí se vypočítá pomocí funkce 11.

$$x_T=(x-x_c)(1+K_1r+K_2r^2+K_2r^3)$$
$$y_T=(y-y_c)(1+K_1r+K_2r^2+K_3r^3)$$
$$r=\sqrt{(x-x_c)^2+(y-y_c)^2}$$
Kde x_c je střed zkreslení
K_1, K_2, K_3 jsou koeficienty zkreslení

Vzorec 11: Transformace bodu

Vypočítané koeficienty K jsou charakteristické pro konkrétní objektiv, stačí nám tedy tyto koeficienty vypočítat pouze jednou a na další snímky aplikovat funkci již bez výpočtu koeficientů. Pokud bychom chtěli korekci dalších geometrických zkreslení (např. Tangenciální), potom bychom patřičným způsobem modifikovali vzorec 11.

3.Vlastní práce

Program pro kalibraci je napsán v programovacím jazyku C (norma ANSI C99), který je lehce portovatelný na různé platformy, jednočipy atd. V programu je použita pro některé základní úkony volně šiřitelná knihovna openCV (z anglického slova Open Source Computer Vision). Program je psán v prostředí Linuxu a je kompilován překladačem gcc.

3.1.Předzpracování obrazu

V této části programu se budeme snažit upravit vstupní obraz tak, abychom mohli co nejlehčeji detekovat přímkové úseky. Nejdříve rozdělíme barevný obraz na jednotlivé barvy, podle obrázku č.8. Vyhledáme hrany a odfiltrujeme vzniklý šum. Poté obrazy s hranami spojíme do jednoho, který bude takto připraven na segmentaci.

3.1.1.Načtení a rozložení obrazu na barvy

Zkoumaný snímek načteme pomocí knihovny openCV a funkce *cvLoadImage()*, která si poradí s formáty patřící do mezinárodního standartu (ISO96) (JPEG, PNG, TIFF), ale i s formáty jinými, jako například BMP, DIB ,PPM, RAS. Načtený obrázek je ve struktuře IplImage, kde jsou uloženy kromě obrazových dat i všechny potřebné informace o obrázku. Nejdůležitější komponenty této struktury pro program jsou:

- *int* ***nChannels*** – počet kanálů v obrázku(1-4),
- *int* ***width,*** *int* ***height*** - čísla vyjadřují počet pixelů šířky respektive výšky obrazu,
- *char* ****** ***imageData*** – v tomto poli je uložena jasová hodnota každého pixelu. Pole má velikost součinu šířky, výšky a počtu kanálů.

Po úspěšném načtení obrázku následuje jeho rozdělení podle počtu kanálů na obrázky s pouze jedním kanálem. V naší práci budeme převážně používat barevný obraz ve formátu *jpg*,

který má tři kanály, kde každý kanál představuje jednu barvu z barevného modelu RGB (kapitola 2.2.), pak vytvoříme právě tři obrazy, kde každý obraz bude v odstínu šedi vyjadřovat barevné zastoupení (obrázek 8). V originálním barevném obrázku jsou data v poli *imageData* uspořádána tak, že každému pixelu náleží tři po sobě jdoucí jasové hodnoty pole v pořadí: modrá, zelená, červená. Separování barev je tedy velmi jednoduchou úlohou.

Obrázek 8: Rozdělení barevného obrázku do složek R,G,B.

3.1.1.Potlačení šumu

Jak jsme si ukázali v kapitole 2.3.1., šum se dá odstranit několika způsoby. Většinou dochází k rozmazání ostrých hran tím více, čím větší použijeme konvoluční masku. Velikost masky také ovlivňuje účinnost odstranění šumu. Jako kompromis zvolíme malou masku o velikosti 3x3, abychom hrany rozmazali co nejméně. Na obrázku 9 můžeme vidět výsledky odstranění šumu metodou průměrování a metodou mediánu. Originální obrázek 9A byl uměle zašumněn.

Obrázek 9: A - originální obrázek, B - metoda průměrování, C - metoda mediánu.

Jak můžeme vidět, metody se od sebe na první pohled moc neliší. Pokud zkusíme vyhledat přímky, lepších výsledků v tomto případě dostaneme z obrázku 9-B, na který byla aplikována metoda obyčejného průměrování. Použití obou metod musíme vyzkoušet. Můžeme použít i větší rozměr masky, pokud je vstupní obraz velkého rozlišení. Pro náš úkol jsou však většinou vhodnější masky menšího rozměru.

3.1.2.Vyhledání hran v obraze

K vyhledání hran použijeme aproximaci druhé derivace Δ^2. Pokud budeme uvažovat jen jednu osu, tak pomocí rovnice 4 odvodíme aproximaci hranového operátoru tak, jak je naznačeno na obrázku 10. Pro druhou osu je výsledek stejný, ale pootočený. Pokud vytvoříme z těchto výsledků dvě matice 3x3 a provedeme s nimi operace sčítání, odčítání a násobení, dostaneme tři různé masky, které jsou naznačeny na obrázku 11.

$$\Delta_x^2 f_{i,j} = \Delta_x(\Delta_x f_{i,j}) = f_{i,j} - f_{i-1,j} - (f_{i-1,j} - f_{i-2,j}) = f_{i,j} - 2f_{i-1,j} + f_{i-2,j}$$

Vzorec 12: Odvození hranového operátoru pro osu x

I-2,j	I–1,j	I,j
1	-2	1

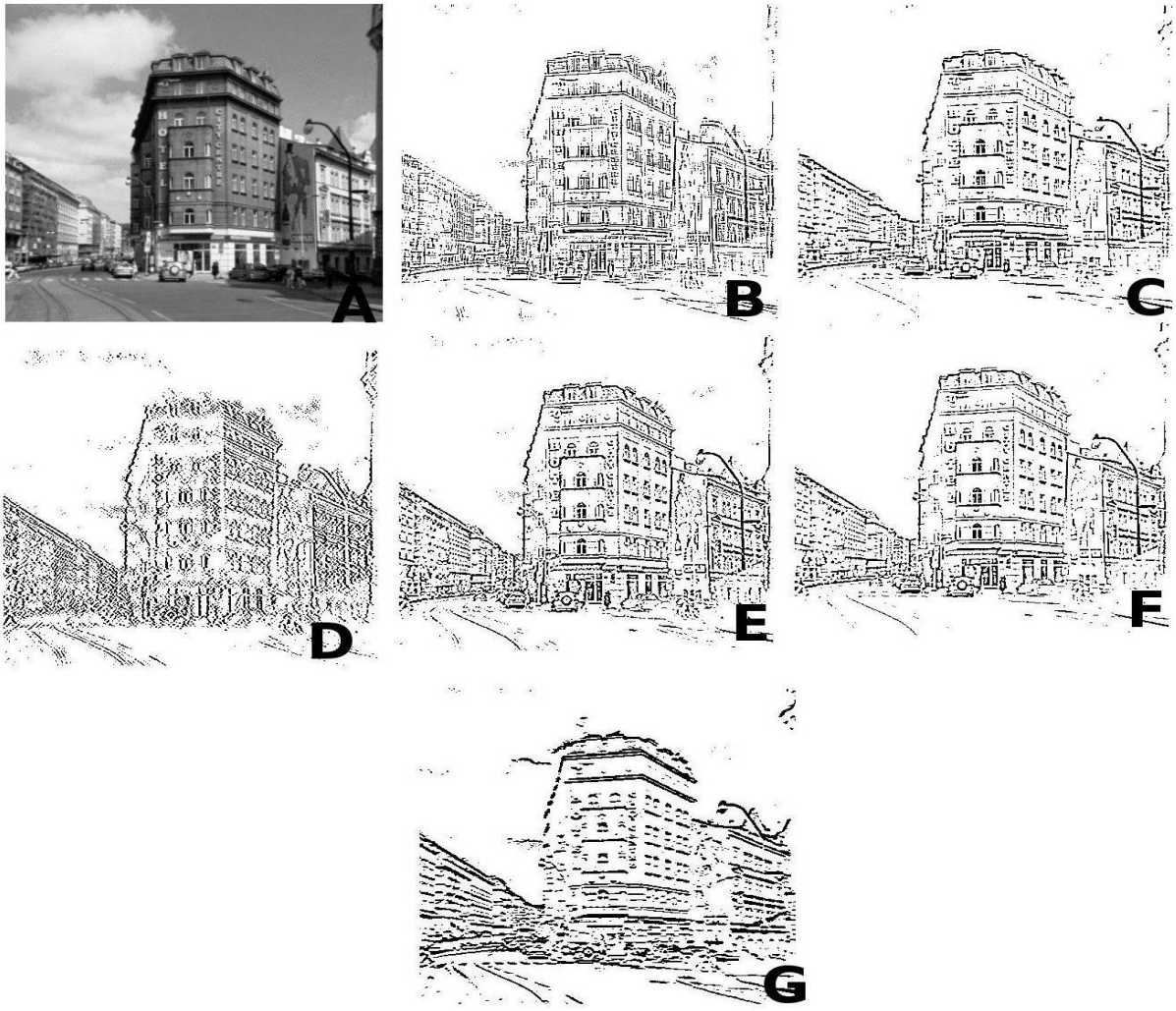
Obrázek 10: Hranový operátor pro osu x.

$$\begin{bmatrix} 0 & -3 & 0 \\ 3 & 0 & 3 \\ 0 & -3 & 0 \end{bmatrix}_1 \qquad \begin{bmatrix} 3 & -6 & 3 \\ -6 & 12 & -6 \\ 3 & -6 & 3 \end{bmatrix}_2 \qquad \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}_3$$

Obrázek 11: Odvozené operátory aproximující druhou derivaci.

Jak je uvedeno v kapitole 2.3.2., součet koeficientů masek by měl být roven nule, což splňují všechny odvozené masky. Prahování provádíme s pevným prahem (výsledek prahování je binární obraz), který určíme experimentálně. Výsledky prahování s operátory (obrázky 3, 4, 5) a s odvozenými operátory (obrázek 11) jsou uvedeny na obrázku 12. Obrázek 12-D je oproti ostatním nepoužitelný, hrany jsou porušeny a rozmazány, obrázek byl hranován odvozeným operátorem 11-2, který tedy není vhodný. Naproti tomu odvozený operátor 11-3 podává velmi dobré výsledky (obrázek 12-C; většina hran je neporušená, obrázek není zašuměn), tento odvozený operátor je spolu s operátorem 5-B vhodný.

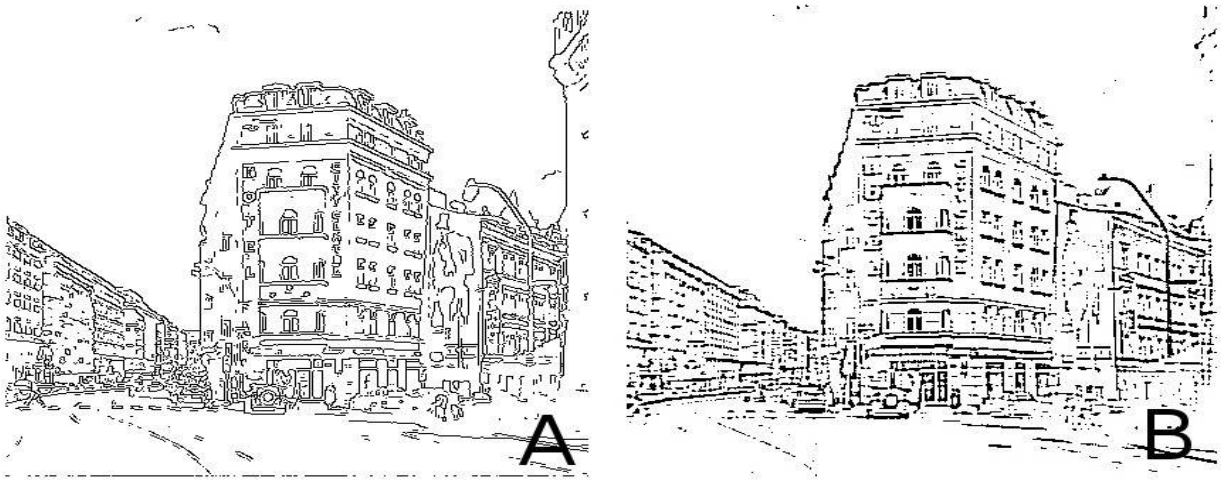
Obrázek hranovaný Sobelovým operátorem (obrázek 12-G), který aproximuje první derivace, má rozmazané hrany, ale není tak zašuměn. Pro další zpracování tento operátor používat nebudeme. Důvodem je, že hrany, které jsou blízko u sebe tento operátor spojí, některé hrany přeruší. Jednodušší je vypořádat se se šumem, než s tímto problémem.



Obrázek 12: A-originální obrázek, B - D odvozený filtr (1, 3, 2), E - Laplaceův filtr, F - Laplaceův filtr, G - Sobelův filtr.

Lepších výsledků, než jsou na obrázku 12, bychom dosáhli pomocí hranování s hysterezí, kterého využívá například Cannyho hranový detektor, který je obecně považován za optimální detektor. Rozdíl mezi Cannyho hranovým detektorem (použitá funkce *cvCanny*) a odvozeným operátorem (obrázek 12-C) je vidět na obrázku 13. Ve vlastním programu budeme tedy pro hranování používat defaultně nastavenou funkci *cvCanny*, ostatní zmíněné filtry si můžeme zvolit za běhu a rozhodnout se, který filtr je nejvhodnější. Hranovací prahy nastavíme také až za běhu, kde se nám bude zobrazovat vyhranovaný obrázek s danými prahy. Tyto prahy budeme měnit tak dlouho, dokud se nám nepodaří najít optimální poměr mezi tloušťkou hran, přerušeností hran

a šumem. V ideálním případě dosáhneme minimálně zašumněného obrazu s tenkými hranami, které nebudou přerušeny. Právě přerušené hrany nám mohou v dalším zpracování dělat problémy.



Obrázek 13: A - hranováno pomocí funkce cvCanny, B - hranováno pomocí odvozeného operátoru 3.

3.1.3.Dilatace a eroze

Po průchodu obrázku hranovým detektorem, se často stane, že mezi souvislými hranami vznikne nespojitost. Tuto nespojitost se pokusíme odstranit dilatací (kapitola 2.3.3.a)). K tomu použijeme funkci z knihovny openCV cvDilate(), jako strukturní element ponecháme defaultní (3x3). Následně provedeme duální transformaci, tedy erozi (kapitola 2.3.3.b)), opět použijeme funkci z knihovny, tentokrát s názvem cvErode(). Výsledky můžeme vidět na obrázku 14. Tento postup je vhodný zejména proto, že vstupní obrázek můžeme hranovat s větší hodnotou prahu, tudíž obrázek je méně zašumněn, roztržené hrany pak částečně spojíme erozí.

Obrázek 14: A – vstupní obrázek, B – Eroze, C – Dilatace.

3.2.Zpracování

Naším úkolem je pokusit se najít ve zkresleném obrázku s neznámým obsahem zkreslené přímkové úseky. Tento úkol nám stěžují jednak obecné problémy, šum, špatná detekce hran, přerušené hrany a pak problémy, závisející na konkrétním případě a úsudku, například to, co ještě budeme považovat za zkreslenou přímku, minimální velikost přímky atd. Některé uvedené problémy částečně řeší předzpracování, bohužel ne všechny a ne zcela. Je tedy zřejmé, že výsledný program nebude univerzálním.

3.2.1.Segmentace přímek

Správná detekce zkreslených přímek je nejtěžší, ale zároveň nejdůležitější úkol. Závisí na něm celá kalibrace, pro kterou bychom potřebovali co možná nejdelší detekované přímky, v ideálním případě přes celý obraz.

Pro segmentaci existuje spousta metod. Vzhledem k časové náročnosti programování použijeme pouze jednu metodu, která bude vyhledávat přímky na základě lokální analýzy nalezených bodů.

Segmentaci budeme provádět podobně, jak je popsáno v kapitole 2.4.2.a) s tím rozdílem, že nevyhledáváme v obrázku hranici objektu. Procházíme tedy binární obraz, dokud nenalezneme

obrazový bod. Ten může náležet hledané přímce. Poté vyšetříme okolí 3x3 bodu, pokud se v tomto okolí vyskytuje jeden okolní bod, pak je součástí aktuální přímky. Takto postupujeme iterativním postupem dále.

Pokud se v okolí vyskytuje bodů více a aktuální přímku tvoří zatím jen jeden bod, vybereme z okolí jeden z nich. Tento postup opakujeme, dokud nezískáme dostatečný počet bodů (např. 10).

Po nashromáždění těchto bodů vypočteme průměrný přírůstek na obou osách a vytvoříme pole posunu, které charakterizuje způsob vyhledání dalších bodů v závislosti na směru přímky. Pole vyplníme podle vypočteného přírůstku a jeho modifikací.

Podle velikosti pole můžeme prohledávat i vzdálené okolí od aktuálního bodu. Podle posunu určeného v poli, budeme prohledávat daný obraz. Pokud v něm najdeme zaplněný pixel, uložíme si jeho pozici, protože se jedná o potenciální další bod přímky.

O tom, který z nalezených bodů patří k přímce, rozhodne funkce, která ohodnotí každý potenciální bod číslem, vyjadřujícím věrohodnost bodu k dané přímce. Hodnotící funkce bere v úvahu směr aktuální přímky a body, které leží v okolí hodnoceného bodu. Hodnocenému bodu se zvedá jeho skóre za každý bod, který v jeho okolí přesně kopíruje aktuální přímku.

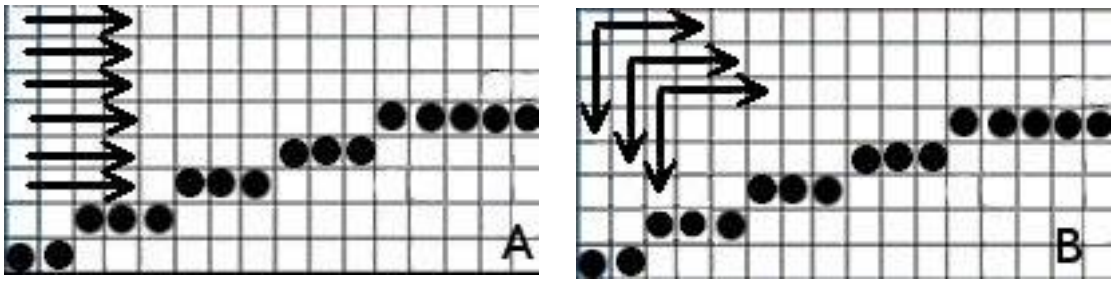
Hodnotící matice, podle které se skóre zvedá, může být například následující (obrázek č. 15). Tuto matici pak orotujeme tak, aby odpovídala směru aktuální přímky. Bod, který má nejvyšší hodnocení, které je zároveň větší než nadefinovaná hranice určená odhadem (např. 8), bude přidán k aktuální přímce. Tím se ukončí jeden cyklus výběru bodu.

1	1	1	1	2	3
1	1	1	2	3	4
X	1	2	3	4	5
1	1	1	2	3	4
1	1	1	1	2	3

Obrázek 15: Vypočet skóre (X – hodnocený bod).

Pokud budeme procházet obraz po řádcích, velmi často se nám stane, že algoritmus na vyhledání zkreslených přímek nám zkreslenou přímku rozdělí tak, jak je ukázáno na obrázku

16A. Proto je výhodné procházení zvolit střídavě 16B, jednou po řádku, jednou po sloupci. Detekované přímky jsou delší, aniž bychom je museli spojovat.



Obrázek 16: Procházení obrázku.

Uvedený způsob je náchylný na šum, proto před jeho použitím odstraníme z obrazu malé skupinky izolovaných bodů. Zmenší se tedy počet potenciálních přímek. Pokud se v detekovaných přímkách nachází velké množství přerušených přímek, pokusíme se je spojit v dalším kroku (kapitola 3.2.2.).

3.2.2. Spojování

Pokud se v obrázku vyskytne přerušená hrana, nalezneme dvě přímky, které potřebujeme spojit. Procházíme přímku po přímce, u aktuální přímky vypočítáme směrnici, a prohledáme všechny ostatní přímky se stejnou nebo podobnou směrnicí, které leží v daném okolí počátečního nebo koncového bodu aktuální přímky.

Výpočet směrnice můžeme provádět přes všechny body, to znamená, že budeme počítat pozici mezi všemi sousedními body na přímce (například pokud je bod vedle aktuálního bodu, přičítáme 1, pokud nahoře, přičítáme 3 atd.). Takový výpočet je poměrně zdlouhavý a stává se, že se spojí i přímky s velmi odlišným tvarem, ale podobnou směrnicí.

Další možností je směrnici vypočíst pomocí počátečního a koncového bodu a bodu uprostřed přímky. Tato metoda je znatelně rychlejší a výsledky jsou srovnatelné.

Další možností výpočtu směrnice je použití metody nejmenších čtverců. Tato metoda se osvědčila jako nejlépe použitelná.

Detekované přímky je vhodné si uložit do souboru, pro práci je to velmi výhodné. Pokud např. chceme použít pro kalibraci více snímků, pak stačí vždy přímky uložit na konec souboru. K tomu slouží funkce *uloz_primku*, tato funkce je přetížená, buď jí jako parametr předáme strukturu *polePrimek*, nebo matici *int***,* další parametry jsou jméno souboru a mód (přepiš soubor nebo přidej na konec existujícího). Na každém řádku souboru je jedna přímka, která je uvozena znakem *, za tímto znakem následují již souřadnice *x, y.* Před souřadnicí *x* je vždy znak @, před souřadnicí *y* je vždy znak *;*.

Pro načítání přímek ze souboru pak slouží funkce *nacti_primky_ze_souboru*, tato funkce vrací strukturu *polePRIMEK*, a jako parametr do ní vkládáme jméno souboru.

3.2.3.Rovnání

Pokud budeme předpokládat, že vstupní obraz má zkreslení buď typu soudek nebo poduška (obrázek 7), můžeme použít vzorec 11 k jeho vyrovnání. Neznámé koeficienty K_1, K_2 a K_3, jsou zkreslení, které musíme určit. Pro určení využijeme nalezené přímky v obraze (kapitola 3.2.1.). Pokud by byly nalezené přímky rovné (nezkreslené), pak je jejich rovnice uvedená ve vzorci 13. Pokud do této rovnice budeme postupně dosazovat body ležící na přímce, bude nám *E* vycházet rovno nule.

$$y = kx + q$$
$$E = y - (kx + q)$$

Vzorec 13: rovnice přímky

Pokud ale budeme do této rovnice dosazovat body vzdálené od přímky, bude nám *E* vycházet různé od nuly. Je jasné, že čím více se bude *E* blížit nule, tím blíže k přímce se nachází bod. Znaménko chyby *E* může být kladné i záporné. Záleží na tom, jestli daný bod je pod nebo na přímce. Pro naši potřebu je to úplně jedno, zajímá nás pouze velikost, proto u výpočtu budeme počítat s absolutní hodnotou. To se však ukázalo jako méně vhodné, lépe se osvědčilo použít

druhou mocninu chyby *E*, která navíc zohledňuje velikost chyby. Čím je chyba větší tím bude mít větší váhu.

Budeme-li mít zkreslenou přímku určenou množinou bodů, pak celkovou chybu od nezkreslené přímky můžeme vypočítat podle vzorce 14.

$$E = \sum_{n=i}^{n=0} (y_n - (k x_n + q))^2$$
$$i = \textit{je pocet bodů}$$

Vzorec 14:Výpočet chyby přímky

Vzorec můžeme rozšířit o předpokládanou korekci zkreslení (vzorce 11), v kterém budeme předpokládat, že střed zkreslení je uprostřed obrazu tedy v bodě [0;0].

$$E = \sum_{n=0}^{n=i} ((y_n - y_c)(1 + \sum_{j=1}^{3} K_j r^j) - (k(x_n - x_c)(1 + \sum_{j=1}^{3} K_j r^j) + q))^2$$

Vzorec 15:Výpočet chyby přímky při korekci

Nalezení minima nám komplikuje skutečnost, že u detekovaných přímek neznáme ani jednu z konstant přímky *k, q*. Neznámé parametry v rovnici jsou tedy *k, q, K1 ,K2 ,K3*. K nalezení minima funkce více proměnných existuje několik teoretických postupů. V programu můžeme použít nejjednodušší způsob, a to takový, že vždy změníme jeden parametr o určitou kladnou hodnotu a spočítáme sumu. Pokud se suma zvětší, zkusíme parametr změnit o zápornou hodnotu. Pokud se suma opět zvětší, necháme parametr na původní hodnotě. Pokud se suma alespoň v jednom případě sníží, budeme měnit parametr o určitou hodnotu tak dlouho, dokud se suma nezačne zvyšovat. Takto budeme postupovat s každým neznámým parametrem.

Pro rovnání je vhodné využít více přímek, proto vzorec 15 rozšíříme o jednu sumu, která bude chybu každé přímky sčítat. Zjednodušený vzorec může vypadat takto (vzorec 16), kde Q je chyba způsobená všemi přímkami.

$$Q=\sum_{přímky}\sum_{body}(y*kor-(k*x*kor*r^3)+q)^2$$

$$kor=1+K_1 r+K_2 r^2+K_3$$

Vzorec 16: Korekce obrázku

Rovnici v takovémto tvaru můžeme použít pouze za předpokladu, že detekované přímky nejsou svislé. Svislé přímky mají směrnici *k* limitně jdoucí k nekonečnu, což je nepoužitelné. Tento problém můžeme vyřešit tím, že u přímek s větším sklonem, než je určená hranice, zaměníme souřadnice. Vzorec přímky by tedy mohl vypadat takto (vzorec 17).

$$x=ky+q$$

Vzorec 17:Rovnice přímky

Hranici, od které budeme souřadnice zaměňovat, je vhodné určit s hysterezí. Například pro přímky se směrnicí 0° - 80° použijeme klasické vyjádření rovnice a pro přímky se směrnicí 70°- 110° použijeme rovnici se zaměněnými osami. Znamená to, že přímky v oblasti hystereze použijeme dvakrát, ale s jinou rovnicí. Pakliže směrnice překročí maximální hodnotu vzhledem k úhlu (v našem případě hodnotu 1,77), pak tuto přímku v dalších krocích neuvažujeme.

Dalším řešením je použití rovnice přímky, ve které nebude směrnice *k*. Nejlepší volbou se zdá být normálová rovnice přímky (viz vzorec 18).

$$x\cos\psi+y\sin\psi-n=0$$

Vzorec 18:Normálová rovnice přímky

Členy $\cos\psi$ a $\sin\psi$ představují složky jednotkového vektoru kolmého k přímce. A *n* představuje velikost kolmice přímky od počátku soustavy souřadnic. Použitím této rovnice nám odpadá úkol sledování směrnic. Můžeme ji použít na všechny přímky stejně, proto budeme preferovat tento způsob. Výsledná kriteriální rovnice pak vypadá takto (vzorec 19).

$$Q = \sum_{přímky} \sum_{body} ((x * kor * \cos\psi) + (y * kor * \sin\psi) - n)^2$$
$$kor = (1 + K_1 r + K_2 r^2 + K_3 r^3)$$

Vzorec 19:Výpočet chyby

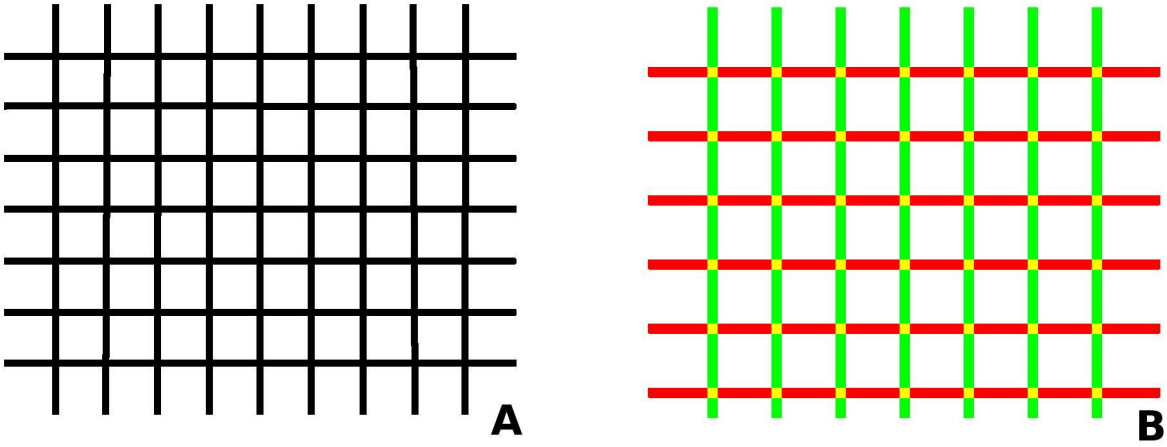
Koeficienty *K* korekce nám vyjdou tím přesněji, čím větší budou mít detekované přímky informaci o zkreslení. Dalším faktorem je volba velikosti kroku inkrementace neznámých koeficientů. Čím menší bude krok, tím přesnějšího výsledku dosáhneme, ale za cenu zvýšení časové náročnosti programu. Od určitého kroku další zlepšení nenastane a výpočetní čas se bude dále zvětšovat.

3.3.Testování programu

Pro otestování algoritmů, které řeší výše popsané postupy, využijeme testovací databázi obrázků. Abychom mohli zkoušet i detekci zkreslených přímek nebo rovnání obrazu, musíme mít k dispozici zkreslený snímek. Proto musíme vstupní obraz zkreslit, pokud ještě není zkreslen. Ke zkreslení využijeme vzorce popsané v kapitole 2.4.4. a vytvoříme funkci pro zkreslení.

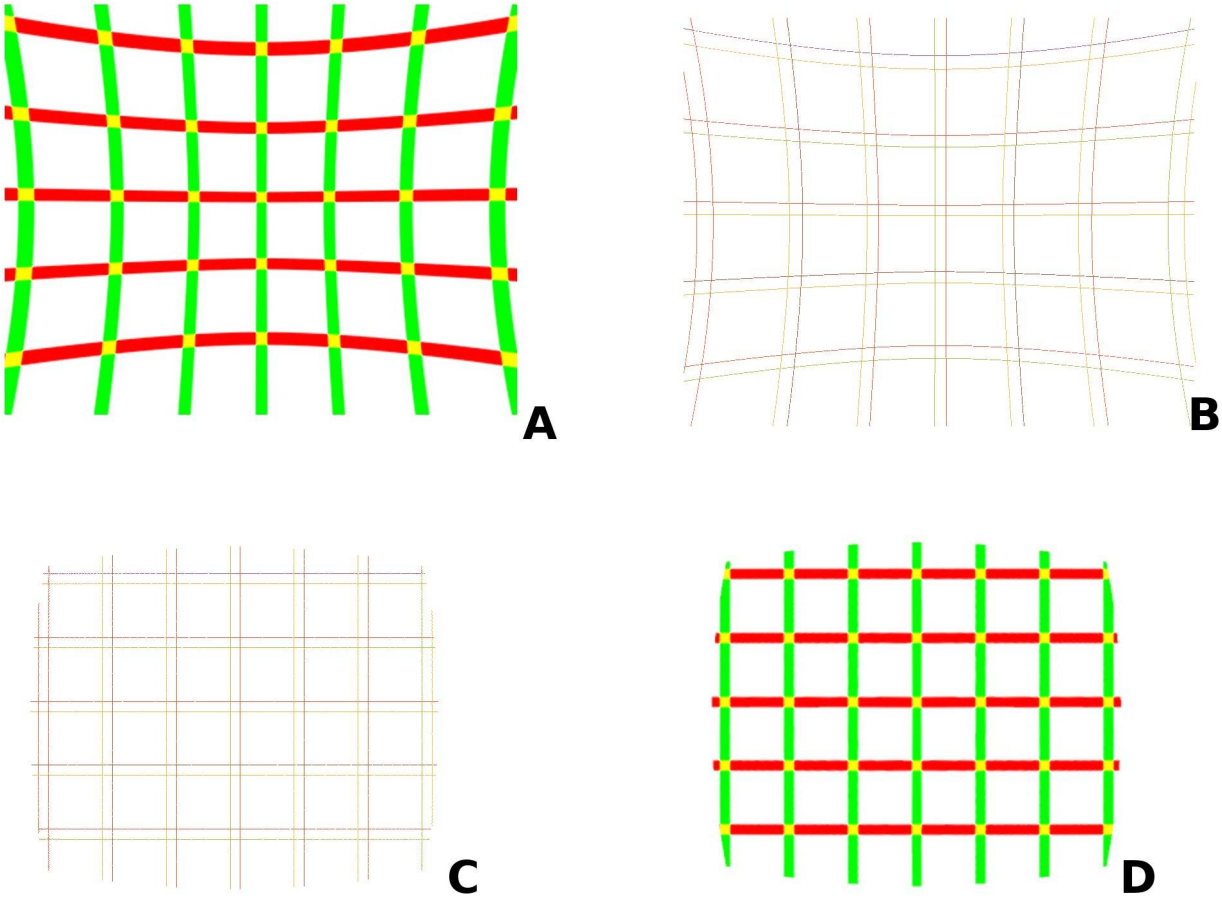
3.3.1.Programem vygenerované kalibry

Tyto kalibry, které si vytváříme sami, slouží pro počáteční testování a ladění algoritmů v průběhu jejich psaní. Na obrázku 17 je mřížka v odstínech šedi a mřížka s barevně oddělenými přímkami, které jsme si vytvořili a budeme je používat jako testovací kalibry. Tyto kalibry je vhodné ukládat nekomprimovaně tak, aby nedocházelo ke ztrátě informací (hlavně k rozmazání barev u hran), vhodný formát může být například *bmp*.



Obrázek 17: Vygenerovaný kalibr A) v odstínech šedi, B) barevný.

Vytvořené algoritmy otestujeme na barevném kalibru, který nejprve zkreslíme. Poté ho rozdělíme na jednotlivé barevné složky a gradientním operátorem nalezneme hrany, ve kterých se pokusíme naleznout zkreslené přímky. V tomto namodelovaném případě nalezne náš algoritmus všechny přímky v plné velikosti (přes celý obraz) a bez přerušení. Bohužel v reálných snímcích tato situace téměř nikdy nenastane (každá detekovaná přímka je pro lepší přehlednost rozlišena barvou). Z nalezených přímek se pokusíme odhadnout koeficienty, kterými byly přímky zkresleny. Pomocí těchto koeficientů můžeme vyrovnat celý obraz. To vše je ukázáno na obrázku č.18.



Obrázek 18: A) Generovaná zkreslená mřížka. B) Nalezené zkreslené přímky. C) Vyrovnané nalezené přímky. D) Vyrovnaný obraz podle nalezených koeficientů.

Ke zkreslení obrazu jsme použili vzorec č. 20 s koeficientem zkreslení *k=0,005*. Ke korekci zkreslení můžeme použít vzorec č. 21. Nalezené koeficienty zkreslení jsou K_1=-4,948e-4 K_2=1,142e-7 a K_3=3,54e-11. Můžeme si tedy všimnout, že náš algoritmus nenašel odpovídající koeficient, jakým jsme obraz zkreslovali. Nepřesnost je způsobená sníženou schopností hledání koeficientů z důvodu úspory času. Pokud bychom přesnost algoritmu zvýšili, nalezený koeficient by byl stejný jako ten, který jsme použili při zkreslení, pouze s opačným znaménkem. Ostatní koeficienty by byly nulové. Pokud srovnáme vyrovnané obrazy, rozdíl je velmi nepatrný.

$$D_{x,y} = S_{x,y} * (1 + k * r) - Zkreslení$$

Vzorec 20: Zkreslení kalibru

$$D_{x,y} = S_{x,y} * (1 + K_1 r + K_2 r^2 + K_3 r^3)$$

Vzorec 21: Korekce kalibru

Narovnaný obraz (18D) je zcela vyrovnán. Chybějící části v rozích obrazu jsou způsobeny zkreslením, které představuje nenahraditelnou ztrátu informace. Pokud by měl objektiv zkreslení stejné, jaké jsme použili na zkreslení mřížky, mohli bychom použít tyto koeficienty na rovnání všech snímků pořízených tímto objektivem, aniž bychom je museli znovu počítat. To ukazuje obrázek č.19. Obrázek byl zkreslen stejným koeficientem jako kalibr a vyrovnán byl již dříve vypočítanými koeficienty. Na vyrovnaném obraze si lze všimnout, že narovnání není úplně přesné tak, jak tomu bylo u kalibru. To je způsobuje právě přesnost vypočítaných koeficientů, které jsme určovali na kalibru, který byl rozměrově asi 3x menší. Pokud tedy budeme počítat koeficienty zkreslení na obraze, který bude menší, než obrazy kalibrované, musíme zpřesnit výpočet. Na snímku je opět vidět část, o kterou jsme přišli zkreslením. Tato část je větší ze stejného důvodu, který byl popsán výše.

Obrázek 19: Zkreslený a narovnaný obraz.

3.3.2.Kalibry nasnímané nezkreslenou optikou

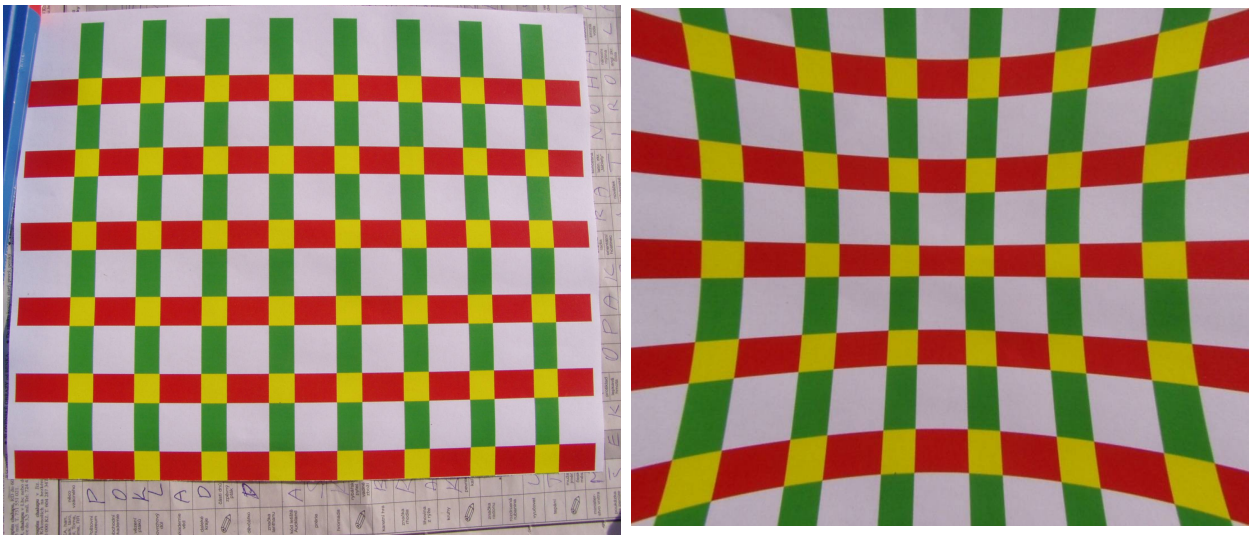
Hlavní problém správného určení parametrů zkreslení je detekování přímek nebo jejich úseků, které nesou co největší informaci o zkreslení. Abychom tyto úseky mohli v nasnímaném obraze naleznout, musíme obraz správně předzpracovat. Předzpracování obrazu je možné všemi postupy, které jsou uvedeny výše. Musíme zde ale zdůraznit, že pro každý snímek je třeba individuálního nastavení parametrů předzpracování a použití jen těch operací, které jsou nutné pro konkrétní snímek. Prostřednictvím těchto kalibrů budeme tedy testovat hlavně algoritmy pro předzpracování obrazu a detekci přímek v reálně vyfocených obrazech (uložených ve formátu *jpg*).

Pro práci použijeme stejné kalibry jako v předchozí kapitole (3.3.1.), které vytiskneme a nasnímáme fotoaparátem. Poté postupujeme stejně s tím rozdílem, že v průběhu předzpracování obrazu vybíráme různé metody a parametry předzpracování podle aktuální situace.

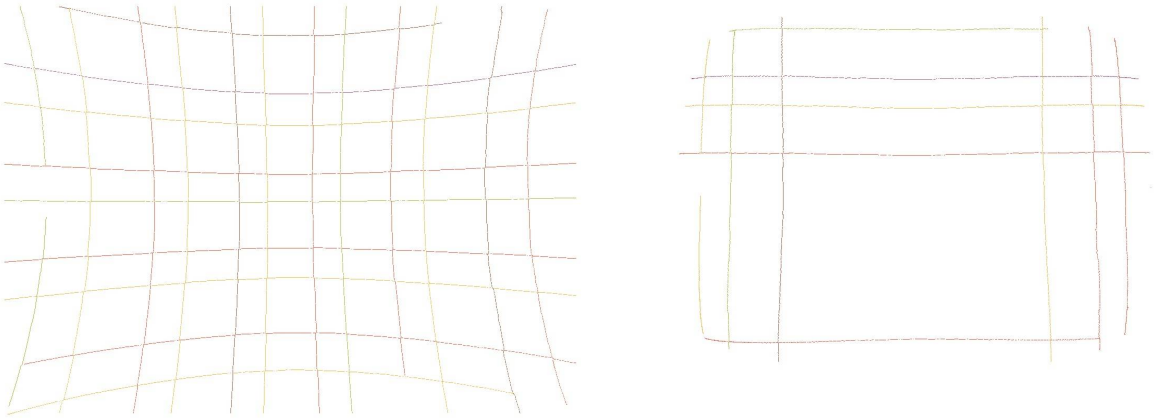
Na obrázku č.20 je zobrazen nasnímaný kalibr a jeho umělé zkreslení typu poduška. Protože zkreslení bylo zvoleno poměrně velké, došlo k osekání původního obrazu. Na dalším obraze (č.21) již vidíme detekované přímkové úseky. Zdá se, že algoritmus našel většinu přímek. Velká část jich je spojitá, některé z nich jsou ale přerušené důsledkem předzpracování. Aby došlo ke spojení těchto úseků přímek, které patří k sobě, můžeme použít algoritmus pro spojování. Výsledné přímky jsou opravdu spojené.

Protože přímek je velké množství, algoritmu jejich zpracování zabere více času. Proto vybereme ze všech detekovaných přímek jen ty, které nesou největší informaci o zkreslení. Tím algoritmus zrychlíme. Pokud nám vadí nepřesnost, která vznikne, můžeme algoritmus pustit ještě jednou, tentokrát se všemi přímkami. Nyní však algoritmus bude vycházet z nenulových počátečních hodnot koeficientů zkreslení.

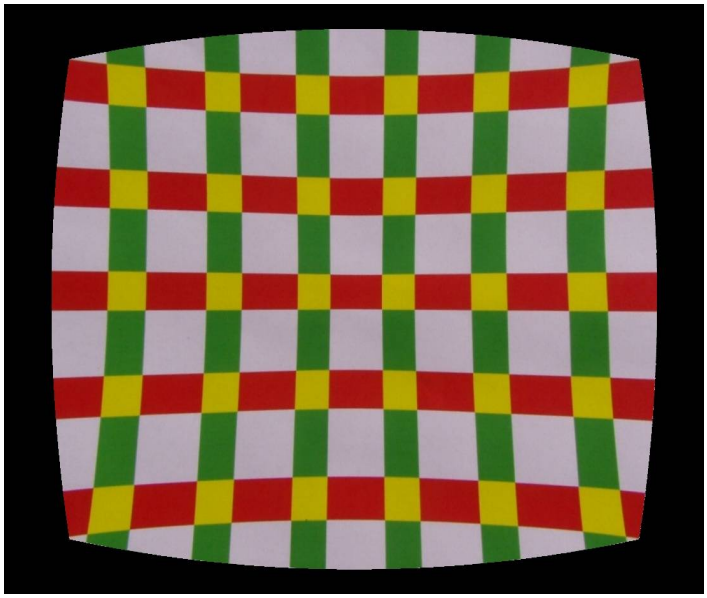
Vyrovnaný obraz si můžeme prohlédnout na obrázku č. 22, kde opět vidíme ztrátu informace v důsledku zkreslení.



Obrázek 20: Vyfocený a zkreslený kalibr formátu A4.



Obrázek 21: Nalezené přímky ve zkresleném obraze, vyrovnané nejdůležitější přímky.



Obrázek 22: Vyrovnaný obraz.

Pokud budeme algoritmy zkoušet na snímcích s neznámým obsahem, nenalezneme dostatečný počet takových přímek, které nesou potřebnou informaci o zkreslení. Proto se nám vyrovnání obrazu nepodaří. Mohou nastat dva možné případy. Buď algoritmus uteče (výsledný

obraz bude více zkreslen a to buď do tvaru soudku nebo podušky, ve většině případů se to dá poznat pouze ze znaménka koeficientů), nebo zůstane beze změny (koeficienty vyjdou nulové). Jistě bychom nalezli snímky s obecným obsahem, ve kterých by program fungoval (na fotce musí být obsaženo co nejvíce souvislých hran, nejlépe přes celý obraz, například fotka panelových domů atd.), ale pro obecnější použití je lépe nasnímat kalibr, určit koeficienty zkreslení a až poté rovnat snímky s neznámým obsahem.

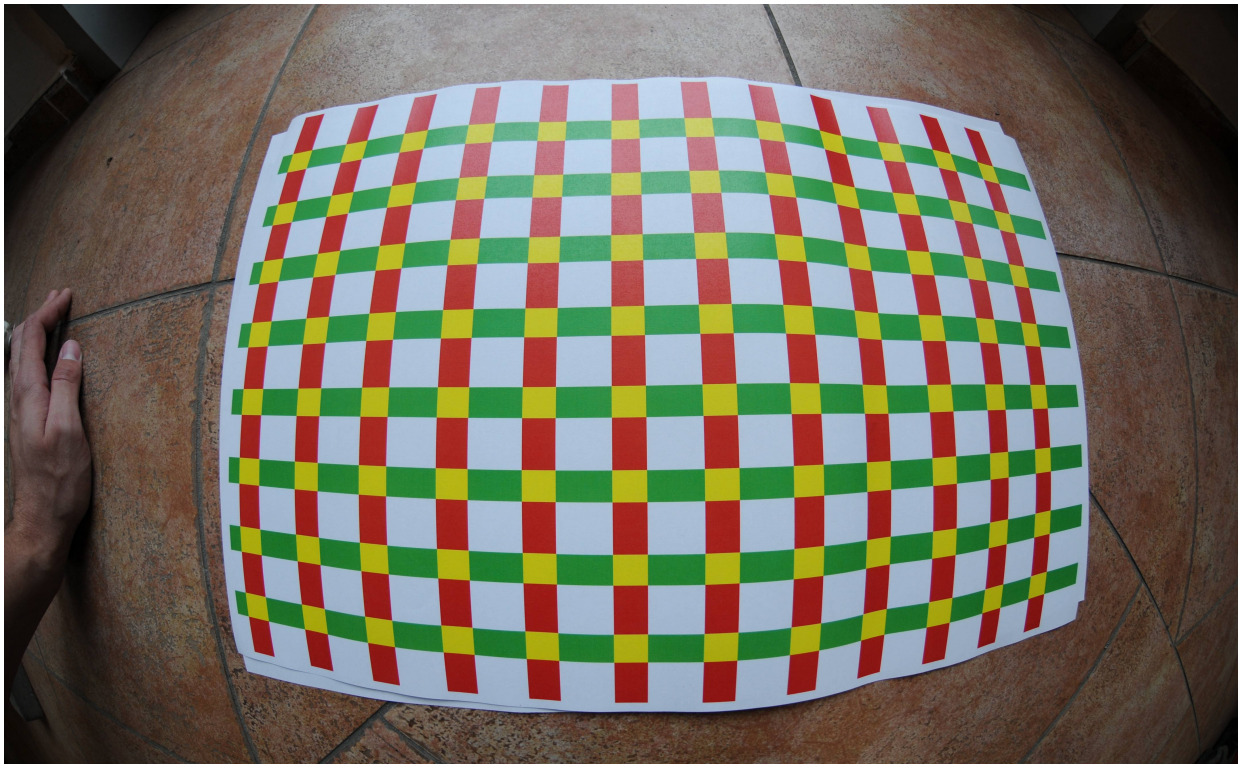
3.3.3.Kalibry nasnímané zkreslenou optikou

V minulých kapitolách (kapitola 3.3.1. a kapitola 3.3.2.), jsme kalibr zkreslovali uměle, až po nasnímaní. Měli jsme tedy k dispozici i nezkreslený obraz. V této kapitole vyzkoušíme náš program na reálném zkreslení optikou.

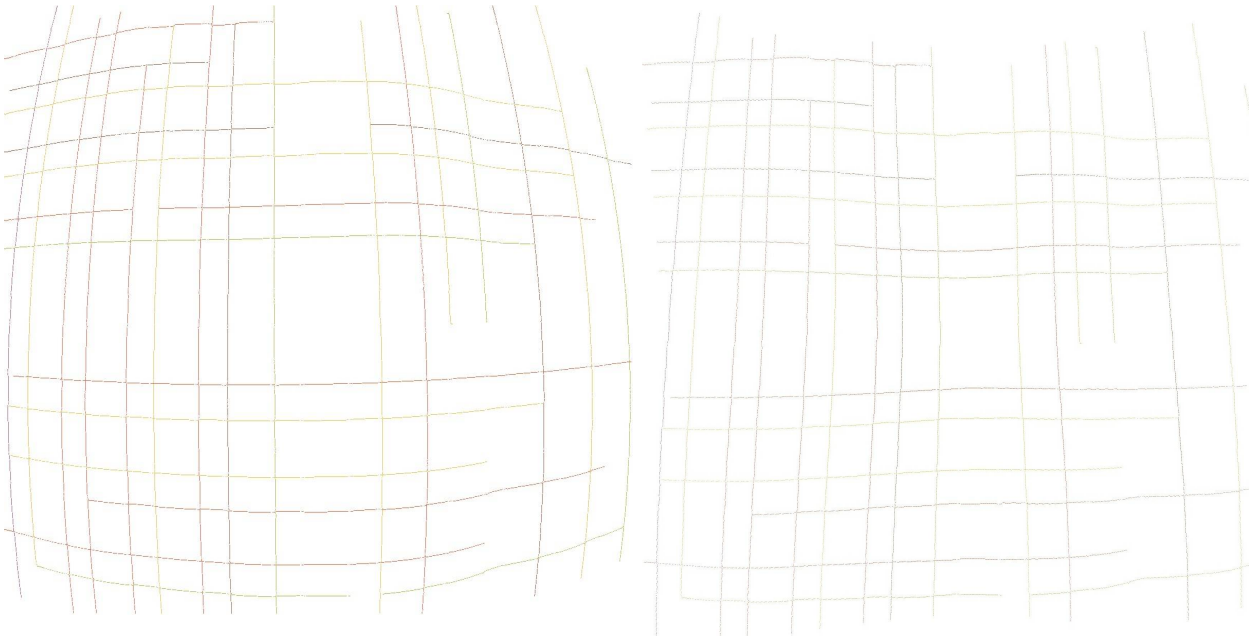
Jednou z možností, jak docílíme reálného zkreslení, je vytvořit obraz pomocí zapůjčeného objektivu Nikon 10,5 mm. Daný objektiv je typu rybí oko a jedná se tedy o soudkové zkreslení (toto zkreslení je pro objektivy asi nejtypičtější). Širokoúhlý objektiv Nikon 10,5 mm má úhel záběru téměř 180°, zkreslení je tedy poměrně velké a v tomto případě záměrné.

Obrázek č. 23 nám ukazuje nasnímaný kalibr (list formátu A3). Zde si také můžeme všimnout, že objektiv má opravdu velký záběr, proto se nám nepodařilo roztáhnout kalibr přes celý obraz, i když byl focen asi z 15 cm výšky. Pokud bychom chtěli, aby se kalibr rozprostíral na větší části snímku, museli bychom buď zvětšit plochu kalibru (např. na list formátu A1 a větší, který ale nebyl k dispozici), nebo snímek vyfotit z menší vzdálenosti, což při daných světelných podmínkách bylo nereálné. Nakonec se však ukázalo, že pro demonstraci námi vytvořeného programu je toto nasnímání dostatečné. Větším problémem je, že kalibr byl při snímání na několika místech mírně zvlněn a odstranění zkreslení je tedy hůře rozpoznatelné.

Na dalším obrázku (č. 24) jsou vidět detekované přímky v celkovém počtu 29, které jsou vhodné pro nalezení koeficientů. Přímky jsou dlouhé, většinou přes celý kalibr, zbylé přímky byly přerušené (krátké), proto jsme je odstranili.

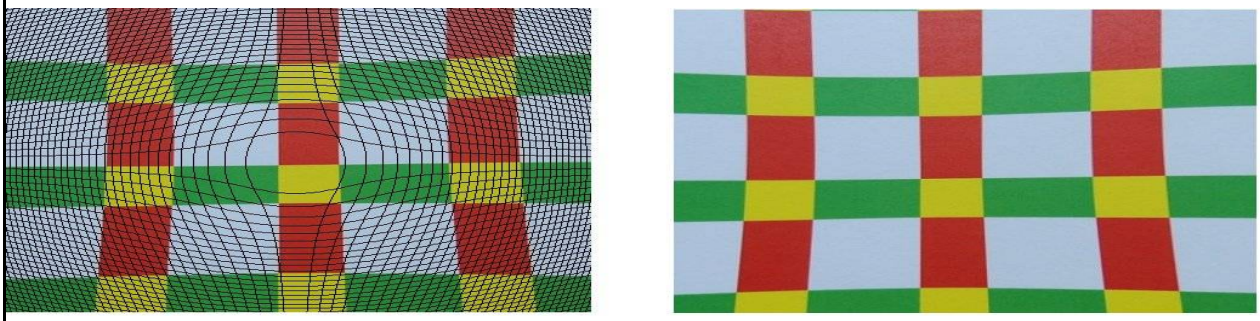


Obrázek 23: Zkreslení objektivem Nicon.



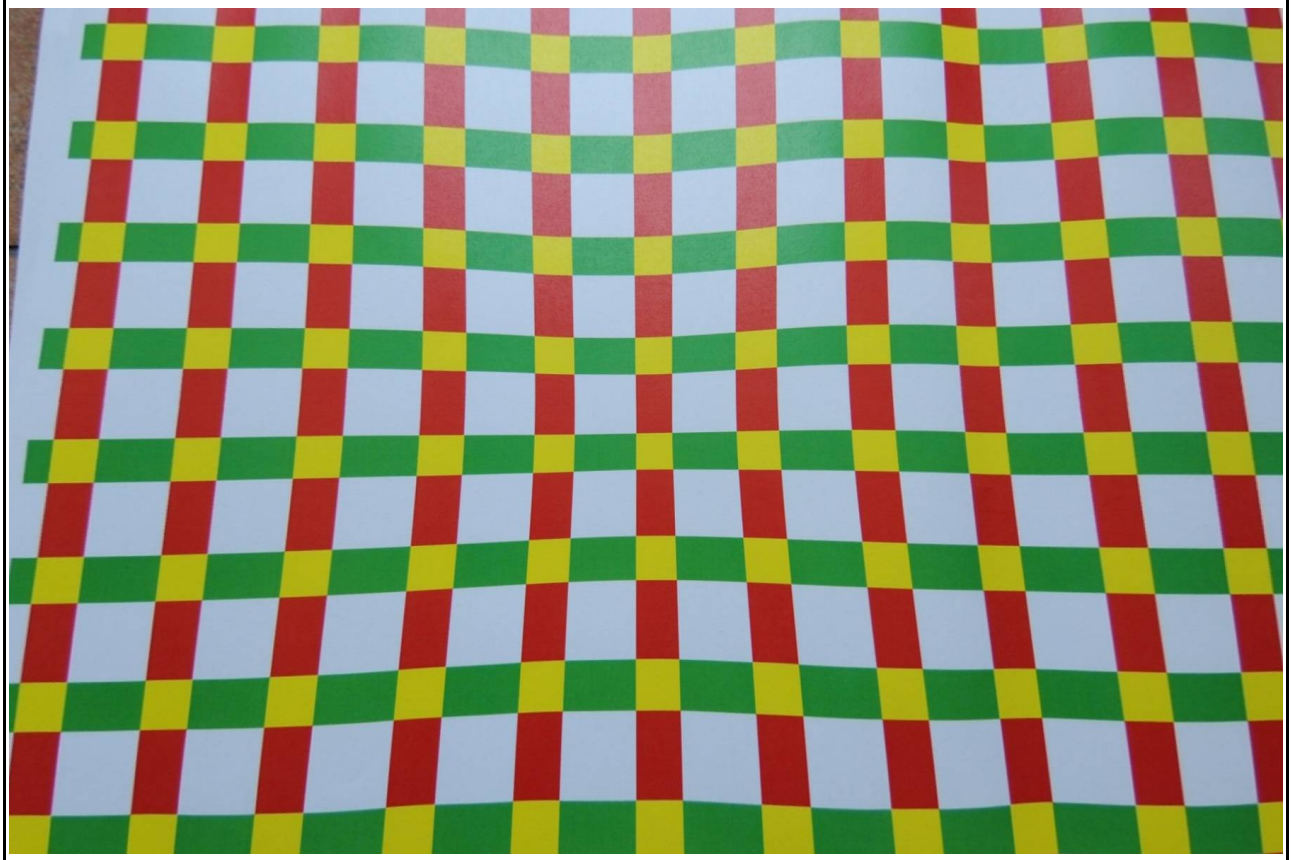
Obrázek 24: Přímky nalezené ve zkresleném obraze optikou Nicon a jejich vyrovnání.

Na výseku vyrovnaného kalibru (obrázek č. 25), se nám objevily rušivé čáry, které na originále nebyly. Tyto čáry naznačují průběh korekce obrazu a jsou důsledkem špatného mapování. Proto použijeme mapování z korigovaného obrazu do původního. Souřadnice vyjdou většinou neceločíselné a výsledný jas se vypočte jako vzdálený průměr jasů okolních bodů, podle vzdáleností od určené polohy. Jak vidíme, tato metoda je již správná.



Obrázek 25: Výřez narovnaného kalibru vyfoceného objektivem Nicon, rybí oko.

Celý narovnaný obraz si můžeme prohlédnout na obrázku 26. Rovnání se povedlo nad očekávání dobře. Velmi rušivě působí vlnky způsobené zahnutím papíru kalibru, ale program si s nimi dokázal poradit. Velké ořezání je způsobeno poměrně velkým zkreslením objektivu a pokud budeme mapovat obraz tak, jak bylo uvedeno výše, rozměr korigovaného a nekorigovaného obrazu musí být stejný, aby nedošlo k přístupu mimo obraz.



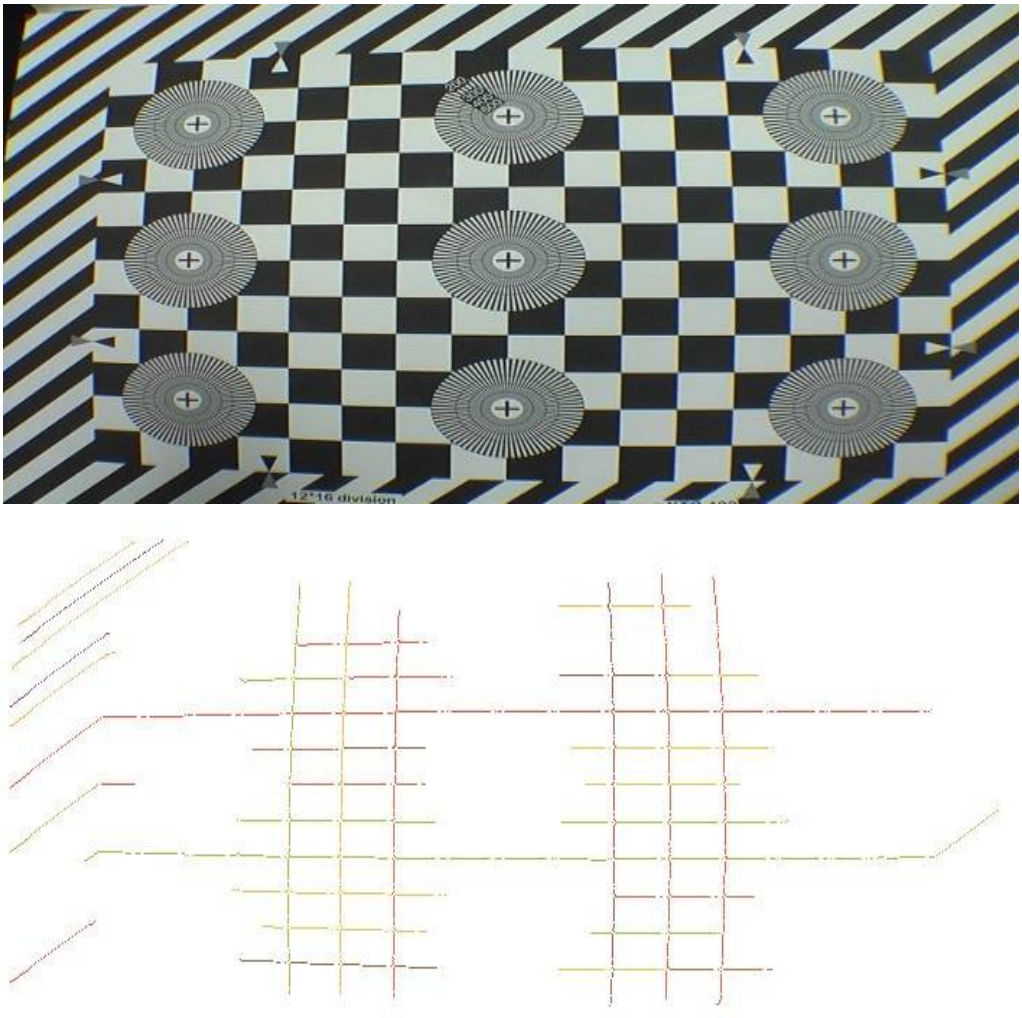
Obrázek 26: Narovnaný kalibr vyfocený objektivem Nicon, rybí oko.

Nyní zkusíme použít vypočtené koeficienty zkreslení na snímek bez kalibru pořízený stejným objektivem. Originální snímek je na obrázku č. 27, na dalším obrázku (č. 28) je již snímek vykorigovaný.

Obrázek 27: Snímek pořízený objektivem Nicon, rybí oko.

Obrázek 28: Narovnaný snímek vyfocený objektivem Nicon, rybí oko.

Další případ reálného zkreslení je průmyslová kamera, s kterou bylo nasnímáno video v rozlišení 704 x 288 pixelů. Z pořízeného videa jsme dostali k dispozici 5 snímků kalibru. Z těchto snímků vybereme jeden, který se nám bude zdát nejvhodnější, a pomocí něho se pokusíme nalézt koeficienty zkreslení (obrázek 29). Na tomto obrázku jsou také zobrazeny nejdůležitější detekované přímky, použité k rovnání. Je vidět, že nejsou moc dlouhé. V nejdůležitějších místech, kde je zkreslení největší (v krajích), byly přímky krátké a proto nevhodné k použití. Samozřejmě bychom je mohli ruční editací souboru s přímkami spojit, ale to by bylo pro reálné použití značně nepraktické. Přímek, které jsou přes celý kalibr, není mnoho. To je způsobeno použitým kalibrem, ve kterém jsou uprostřed „kola“, která přímky protínají. Opět bychom tyto přímky mohli spojit ručně.



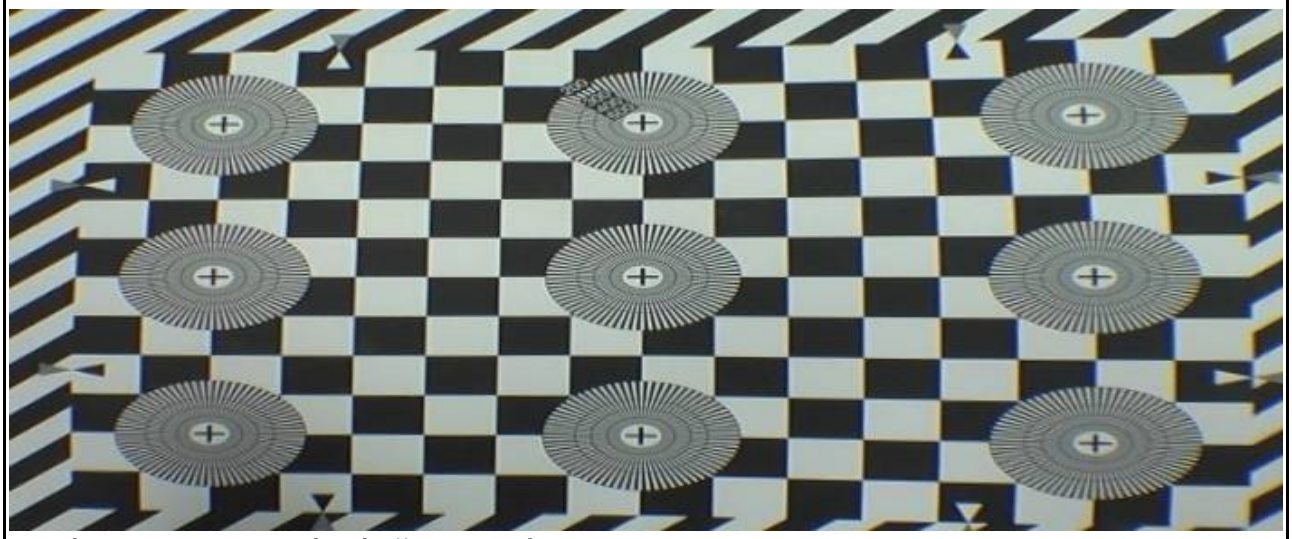
Obrázek 29: Reálné zkreslení kamerou a nejdůležitější detekované přímkové úseky.

Na obrázku č. 30 již máme vyrovnaný kalibr. Vypočítané koeficienty pro kalibraci jsou K_1=2,928e-04, K_2=3,199e-08, K_3=1,00e-13. Jak vidíme na obrázku, ke zlepšení došlo, bohužel ne k tak výraznému, jako v jiných případech. Je to způsobeno právě nedostatkem informací o zkreslení v detekovaných přímkách. Určitě by pomohlo, kdyby rozlišení videa bylo větší a pokud možno i použití jiného druhu kalibru. Ani jednu možnost jsme bohužel nemohli ovlivnit.

Mohli bychom použít nejkvalitnější (vzhledem k informaci o zkreslení) přímky ze všech kalibrů, protože jsou zkresleny stejnou optikou. Bohužel v našem případě nemáme moc štěstí

a přímek, které jsme mohli přidat k původním, moc nebylo. Vypočítané koeficienty jsou K_1=2.933e-4, K_2=3.109e-8, K_3=-4e-11.

K tomu, abychom mohli snadněji porovnat rozdíly mezi vypočtenými koeficienty, můžeme použít poměr celkové chyby *Q* (poměr mezi počáteční a koncovou hodnotou Q). Poměr chyb, pokud použijeme přímky z jednoho obrazu, bude Q1 = 3549 a pokud použijeme přímky ze všech obrazů bude poměr Q2 = 3732. Vidíme, že rovnání s přímkami ze všech snímků přineslo určité zlepšení, nicméně toto zlepšení není nijak závratné, protože přímky z ostatních snímku nepřinesli žádnou novou informaci o zkreslení.



Obrázek 30: Narovnaný reálně zkreslený kalibr.

Na narovnaném obraze č. 30 můžeme vidět, že některé úseky už narovnány jsou. U některých došlo ke zlepšení, ale pořád to ještě není ono. Je to z toho důvodu, že zkreslení, které se pokoušíme odstranit, je způsobeno nekvalitou reálného objektivu. Tento objektiv tedy nebude mít stejné zkreslení v celém záběru. K rovnání využíváme aproximační funkci, která se reálnému zkreslení objektivu bude přibližovat více či méně, nikdy ne přesně.

4.Závěr

V této práci jsme se seznámili s různými metodami zpracování obrazu. V praktické části, která byla zaměřena na kalibrování zkreslených snímků, jsme použili některé ze zmíněných metod. Byl vysvětlen princip vytvořených algoritmů.

V části předzpracování byla představena návaznost nejtypičtějších úkonů předzpracování, které jsou vhodné pro řešení úkolu kalibrace pomocí přímkových úseků. Jejich prosté použití však nezaručí kvalitní předzpracování, je zapotřebí vhodně volit kombinaci těchto úkonů se správnými parametry. To vše můžeme provádět experimentálně za běhu programu.

Jedním z hlavních úkolů předzpracování je správný výběr hranovacího filtru s vhodnými hodnotami prahu. Vybírat můžeme z několika námi vytvořených masek nebo použít funkci *CVcanny()* z knihovny openCV. Hodnoty prahu měníme pohodlně za běhu programu. Hodnota prahu by měla být co nejvyšší tak, aby se odfiltroval co největší počet nepotřebných dat, ale za minimálních ztrát na potřebných hranách.

Další úlohou bylo nalezení přímek v předzpracovaném obraze. Detekci přímek provádíme pomocí lokální analýzy aktuálního bodu. Rozhodování, kterým směrem by přímka měla pokračovat, se provádí pomocí tzv. skóre okolních bodů. Tento přístup se oproti jiným poměrně dobře osvědčil. K jeho přednostem patří odolnost proti přerušení přímek a udržování správného směru v oblasti křížení hran a potlačení šumu. Pokud algoritmus pro detekci přímek v důsledku velkého zašumnění nebo velkého přerušení hrany rozdělí jednu přímku do dvou, můžeme použít algoritmus pro spojování.

Rovnání provádíme pomocí rovnice korekce $kor = (1 + \sum_{j=1}^{3} K_j r^j)$, koeficienty K_j nalezneme pomocí kriteriální funkce tvaru $Q = \sum_{přímky} \sum_{body} ((x * kor * \cos\psi) + (y * kor * \sin\psi) - n)^2$.

Na testovací databázi snímků si můžeme prohlédnout činnost algoritmů nejprve na uměle vytvořených kalibrech, potom na kalibrech vyfocených nezkreslenou optikou. Také jsme si ukázali, jak si algoritmus poradí se snímkem vyfoceným objektivem Nicon 10,5 mm fish-eye, s úhlem záběru 180°. Výsledný obraz je vyrovnán, ale díky velkému zkreslení objektivu byl značně ořezán.

Na závěr jsme program testovali na snímku z reálné průmyslové kamery. Nalezené koeficienty pouze částečně vyrovnaly vstupní zkreslený obraz. Z toho tedy můžeme vyvodit, že rovnání reálného kalibru nevyjde nikdy zcela přesně, neboť funkce, využitá při rovnání přímkových úseků, je pouhou aproximací zkreslení reálné optiky.

Program byl napsán kompletně v jazyku C. Bylo použito několik funkcí z knihovny openCV (*cvLoad, cvSave, cvShowImage, cvCanny, cvErode, cvDilate*). Všechny ostatní funkce byly kompletně vytvořeny v rámci této práce. Program při vhodné volbě vstupního obrazu a parametrů předzpracování dokáže nalézt koeficienty zkreslení a tím vyrovnat zkreslený obraz.

5.Použitá literatura

[1] Faugeras O.: Three-Dimensional Computer Vision, The MIT Press 1993.

[2] Hlaváč V., Šonka M.: Počítačové vidění, Grada, Praha 1992. ISBN 80-85424-67-3.

[3] Horák K., Kalová I., Petyovský P., Richter M.: Počítačové vidění, Brno 2008.

[4] Sobotka Z., Sobotka M.: Počítačová analýza a rozpoznávání obrazu, Praha 1990.

[5] Žára J., Beneš B., Sochor J., Felkel P.: Moderní počítačová grafika, Brno 2004. ISBN 80-251-0454-0.

6.Seznam obrázků

7.Seznam vzorců

8.Seznam použitých zkratek

- OH obor hodnot
- D definiční obor
- z spojitá obrazová funkce
- x, y souřadnice bodů
- I diskrétní ekvivalent funkce
- b počet kvantizačních bitů
- M matice pixelů
- ∇ gradient
- ψ směr gradientu
- r vzdálenost přímky od počátku
- ϕ úhel, který svírá kolmice z počátku k dané přímce a osa x